Les Cahiers du GERAD ISSN: 0711–2440

Improved Column Generation Algorithms for the Job Grouping Problem

J. Desrosiers, R. Jans,
Y. Adulyasak

G–2013–26

April 2013

Improved Column Generation Algorithms for the Job Grouping Problem

Jacques Desrosiers
Raf Jans
Yossiri Adulyasak

GERAD & HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada, H3T 2A7

jacques.desrosiers@hec.ca
raf.jans@hec.ca
yossiri.adulyasak@hec.ca

April 2013

Les Cahiers du GERAD
G–2013–26

Abstract: Dantzig-Wolfe reformulation solved by Column Generation is an approach to obtain improved bounds for Mixed Integer Programs. A downside of this approach is that the Column Generation process can be very time consuming due to degeneracy of the master problem and the tailing-off effect. To counter this problem in the Column Generation process, we also perform a number of Lagrangean Relaxation iterations to generate new columns after each time that we solve the master problem. This potentially reduces the number of times we have to solve the master problem. We explore this combination of Column Generation and Lagrangean Relaxation for the Job Grouping Problem. In our test set, we observe that the time spent to solve the master problem is decreased by a factor 4, while the total time spent on solving the subproblems remains approximately equal.

Key Words: Integer Programming, Column Generation, Lagrangean Relaxation, Job Grouping Problem.

1 Introduction

Dantzig-Wolfe decomposition is a reformulation technique aimed at obtaining improved lower bounds for Mixed Integer Programs (MIP), see Lübbecke and Desrosiers (2005). In this paper we consider minimization problems. The main idea is to split up the original formulation into a set of linking and subproblem constraints. Next, the original variables are replaced by a convex combination of the extreme points of the convex hull of the linking constraints for a bounded problem. This typically leads to a master problem (MP) formulation with a very large number of variables. To solve such large formulations in a practical way, Column Generation (CG) is used. Only a small proportion of the variables are needed to find the linear programming (LP) relaxation of the Dantzig-Wolfe reformulation. Column Generation is an iterative process, where we solve subproblems to generate new columns as needed and we resolve the MP with the newly added columns. The coordination between the master and the subproblems is done via the dual prices of MP, which are used in the objective function of the subproblems to calculate the reduced costs of the missing variables. Over many iterations, the value of MP decreases until we obtain the exact Dantzig-Wolfe LP relaxation bound. In the past years, column generation techniques have successfully been applied to a variety of problems such as vehicle routing and crew scheduling problems (Desaulniers et al. 1998), cutting stock problems (Ben Amor et al. 2006, Degraeve and Peeters 2003) and lot sizing problems (Degraeve and Jans 2007).

It is well known that column generation suffers from degeneracy and a tailing off effect. Degeneracy implies that new columns are found and added to the MP, but the LP relaxation value of this master problem does not improve. Tailing off happens towards the end of the column generation process when many columns have to be added in order to find the optimal value. Both effects slow down the overall column generation process. A lot of research has been devoted to accelerating the column generation process. However, only a few studies have used a combination with Lagrangean Relaxation (LR) to speed up the process, whereas it has a much more general applicability.

The focus of this paper is on evaluating the opportunity to improve the column generation process by combing it with Lagrangean relaxation for clustering problems. We use the Job Grouping Problem to explain and illustrate our approach. The aim is to find an algorithm that efficiently integrates both approaches.

2 On the Integration of Column Generation and Lagrangean Relaxation

Many methods exist to obtain good lower bounds besides Dantzig-Wolfe reformulation solved by Column Generation. Another approach that is widely used is Lagrangean Relaxation. In Lagrangean Relaxation, a set of complicating constraints is dualized into the objective function of the original formulation using a set of Lagrangean multipliers. The solution of such a Lagrangean problem, with a specific set of multipliers, provides a lower bound on the optimal value of the original MIP. The Lagrangean Dual Problem (LDP) consists now of finding the set of multipliers that gives us the best lower bound. The Lagrangean multipliers are generally updated in iterative steps with the aim of improving the lower bound. A traditional approach for updating the multipliers is subgradient optimization (Fisher 1981).

The Column Generation approach and the Lagrangean Relaxation approach have both their advantages and disadvantages.

***Advantages of CG*:** CG is an exact method that results in the exact LP relaxation value of the Dantzig-Wolfe reformulation. At each step of the CG, a lower bound on the optimal LP relaxation value of the Dantzig-Wolfe reformulation can be calculated, by subtracting the optimal reduced costs resulting from the latest subproblems from the value of the latest master problem. This lower bound corresponds in fact to the Lagrangean lower bound with the latest set of dual prices used as the Lagrangean multipliers. (see e.g. Lübbecke and Desrosiers 2005). At each iteration when the master problem is solved, a primal LP solution is found which can be used in heuristics to find an upper bound on the original MIP formulation.

***Disadvantages of CG*:** Using the simplex algorithm to solve the master problem leads to dual prices that are extreme solutions, and that do not necessarily give a good representation if alternative dual solutions exist. Therefore, the columns generated using these simplex dual prices lead to degenerate steps in the master

problem, i.e., they do not necessarily improve the objective function value. Resolving the master problem at each step of the Column Generation process can take a lot of time. Moreover tailing off effect occurs at the end of the optimizing process.

***Advantages of LR*:** The subgradient optimization method for updating the Lagrangean multipliers is very attractive from a computational perspective since it is not a very time-consuming calculation. The subgradients obtained at each iteration are indeed valid columns that can be used in the CG approach.

***Disadvantages of LR*:** LR is not an exact approach. In practice, there is no guarantee that the Lagrangean multipliers will converge towards the optimal multipliers, and the Lagrangean Relaxation is usually stopped before the optimal Lagrangean Dual Bound is found. The Lagrangean Relaxation will hence find a lower bound on the optimal value of the original MIP formulation and on the LP value of the Dantzig-Wolfe reformulation. No primal solution is available, and hence it is (more) difficult to construct upper bounds on the original MIP problem. An upper bound on the original MIP is needed for the subgradient optimization procedure. The update process of the multipliers only makes use of the latest subgradient while CG uses all generated ones (at a much higher computational time).

The combined CG-LR method aims to combine the respective strengths of both methods. This combination is possible because there exist some interesting theoretical links between Column Generation and Lagrangean Relaxation, see Lübbecke and Desrosiers (2005). The optimal value of the CG master problem is the same as the optimal value of the Lagrangean Dual Problem. The Dantzig-Wolfe reformulation and the Lagrangean Dual Problem are in fact each other's dual (Geoffrion 1974). The problem that remains to be solved in the Lagrangean approach is exactly the same as the subproblem in the Column Generation approach. Hence, the subgradient solution of the Lagrangean problem can be used as a new column in the Column Generation process. The dual prices resulting from the optimal master problem also give the best Lagrangean bound (Magnanti, Shapiro and Wagner 1976).

The general idea of the combined CG-LR method is to start by solving the master problem, which is initialized with some columns that guarantee feasibility. The dual prices of this master are used in the subproblems to generate new columns. Instead of immediately resolving the master with the new columns, we are first going to generate more columns via Lagrangean Relaxation. Therefore, we take the simplex dual prices as our initial Lagrangean multipliers, and use subgradient optimization to update the multipliers. Using these new multipliers in the objective function of our Lagrangean problems, we generate new columns. These new columns, generated with Lagrange multipliers, are evaluated using the latest simplex dual prices. If the reduced cost of such a column is negative, we add the column to the current master problem. This process of updating the multipliers using subgradient optimization and of generating new columns with the updated multipliers is repeated for a number of steps. Next, the columns are added to the master, and we solve the master problem using a simplex algorithm again, giving us again simplex dual prices. We continue this process until we reach a suitable stopping criterion, which typically is that no columns with a negative reduced cost are found when solving the subproblems using the latest simplex dual prices, or when the difference between the Lagrangean lower bound and the value of the master LP is within a specified tolerance. An overview of this procedure can be found in Figure 1.

The advantage of this combined method is that we do not always have to solve the master problem in order to generate new dual prices, and hence we avoid this time consuming process. Moreover, the multipliers of the Lagrangean relaxation approach generally provide good columns that can be added to the master problem. Because the subgradient optimization process is not very time consuming, we can hence quickly generate new good columns.

The combined CG-LR method has not received much attention in the literature. Only a few papers have proposed such a combination. Barahona and Jensen (1998) discuss an application for a plant location problem. It has further been applied to solve cutting stock problems (Degraeve and Peeters 2003) and lot sizing problems (Degraeve and Jans 2007). A general discussion can be found in Huisman et al. (2005).

Many questions related to this combined process still exist. How many iterations of LR should be done before returning to solving the master problem? Which upper bound should be used in the subgradient

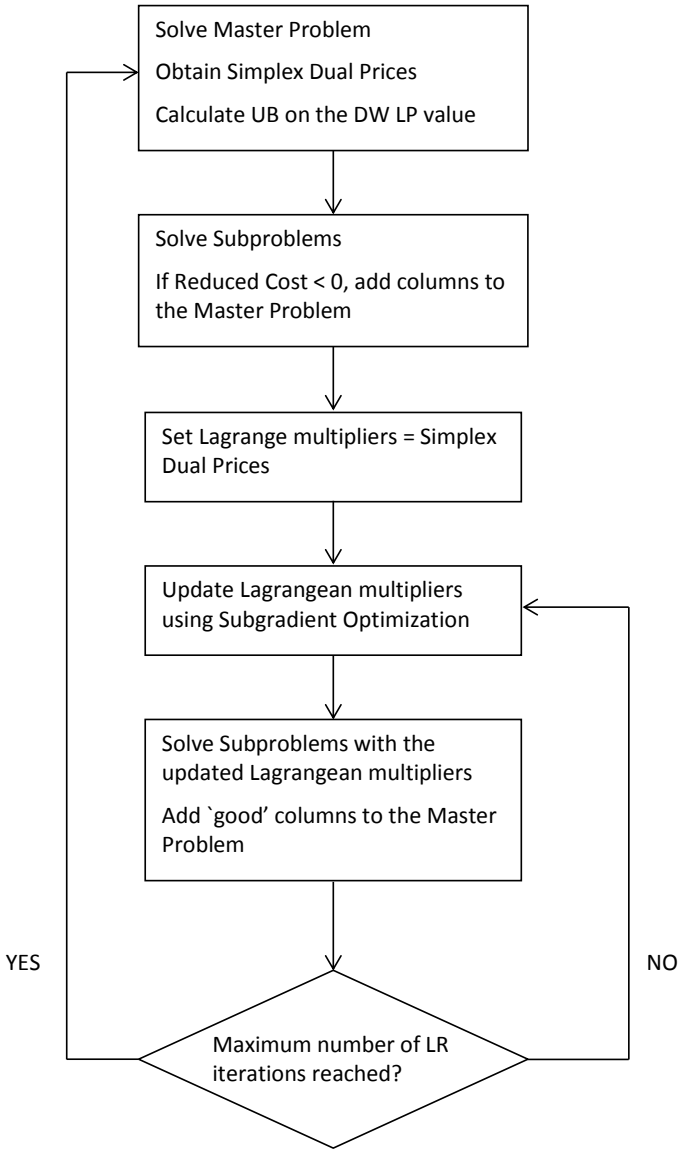

Figure 1: Overview of the combined CG-LR method

updating formula (specifically an upper bound on the original MIP, or the latest master LP value which is an upper bound on the DW LP value). Which columns generated during the LR phase should be added? All of them versus the ones that have a negative reduced cost evaluated using the latest simplex dual prices, or the ones that have a negative reduced cost using the Lagrangean multipliers. After resolving the master problem, should the Lagrangean multipliers be set equal to the latest simplex dual prices, or are there other alternatives? We could set them equal to the last Lagrangean multipliers found, or a combination of the latest simplex dual prices and latest Lagrangean multipliers, or a combination of previous simplex dual prices. What is the impact of the initial columns on this process?

In the following sections, we explore the various issues raised by considering the specific application of the Job Grouping Problem. We discuss in detail how this algorithm can be applied to that problem, and finally we discuss some computational results.

3 The Job Grouping Problem

In the Job Grouping Problem (JGP), several jobs have to be assigned to machines, with the aim of minimizing the number of identical machines used. Each job requires one or more specific tools, which have to be installed on the machine on which the job is processed. Each machine can only hold a limited number of different tools.

To define the problem, we use the following notation:

Sets:

- N the set of jobs $i = \{1, \ldots, n\}$;
- L the set of different tools ℓ;
- L_i the set of tools required to process job i;
- M the set of machines $j = \{1, \ldots, m\}$.

Parameters:

- C maximum number of different tool slots available on each machine;
- s_ℓ number of slots needed by tool ℓ.

Binary decision variables:

- x_{ij} one if job i is assigned to machine j, zero otherwise;
- $y_{\ell j}$ one if tool ℓ is assigned to machine j, zero otherwise;
- z_j one if machine j is used, zero otherwise.

A formulation of the problem is then as follows (Crama and Oerlemans 1994, Denizel 2003, Konak et al. 2008):

$$\text{Min} \quad \sum_{j \in M} z_j \tag{1}$$

$$\text{s.t.} \quad \sum_{j \in M} x_{ij} = 1 \qquad \forall i \in N \tag{2}$$

$$x_{ij} = y_{\ell j} \qquad \forall i \in, \forall \ell \in L_i, \forall j \in M \tag{3}$$

$$\sum_{\ell \in L} s_\ell y_{\ell j} = C z_j \qquad \forall j \in M \tag{4}$$

$$x_{ij}, y_{\ell j}, z_j \in \{0, 1\} \qquad \forall i \in N, \forall \ell \in L, \forall j \in M \tag{5}$$

The objective function (1) minimizes the number of machines used. Each job has to be assigned to exactly one machine (2) and if a job is assigned to a specific machine, then all the tools needed by that job must also be assigned to that machine (3). Finally, each machine has only a limited tool slot capacity (4). All variables are binary (5).

Let v^* be the optimal value of formulation (1)–(5), and let $\bar{v}$ be the LP relaxation value of the same formulation. This LP relaxation value is generally very weak. It has been shown that the LP relaxation is equal to S/C, where $S = \sum_{\ell=1}^{L} s_\ell$ (Tang and Denardo 1988, Crama and Oerlemans 1994, and Denizel 2003). We can make this LP bound a bit stronger by rounding up this value: $\lceil S/C \rceil$. Crama and Oerlemans (1994) proposed to use Column Generation to obtain a better lower bound for the JGP, whereas Denizel (2003) uses Lagrangean Relaxation. In this paper, we look how we can speed up the CG process by combining the two methods.

3.1 Dantzig-Wolfe reformulation and Column Generation for the JGP

We can apply the Dantzig-Wolfe decomposition principle to formulation (1)–(5) by considering the job assignment constraint (2) as the complicated constraint, while constraints (3)–(5) define the domain of the

feasible columns. The general idea of Dantzig-Wolfe decomposition for binary problem is to consider the set of all extreme points, defined by constraints (3)–(5). Let P be this set. We can think of P as the set of all feasible assignments of jobs to a machine so that the tool capacity limit is not exceeded. We also refer to these feasible assignments as columns. In order to formally define the Dantzig-Wolfe reformulation, define a new parameter a_{ki} which equals one if extreme point (i.e. column or assignment) k contains job i. A new binary decision variable s_k indicates if assignment k is chosen or not. The Dantzig-Wolfe reformulation then becomes (Crama and Oerlemans 1994):

$$\text{Min} \quad \sum_{k \in P} s_k \tag{6}$$

$$\text{s.t.} \quad \sum_{k \in P} a_{ik} s_k = 1 \qquad \forall i \in N \tag{7}$$

$$s_k \in \{0,1\} \qquad \forall k \in P \tag{8}$$

The objective (6) minimizes the number of chosen assignments, whereas (7) indicates that each job must appear in exactly one assignment. Given the objective (6) and the fact that each subset of a column is also a valid assignment, we can transform the set of constraints in (7) into greater or equal inequalities:

$$\text{s.t.} \quad \sum_{k \in P} a_{ik} s_k \geq 1 \qquad \forall i \in N \tag{9}$$

In that case, the simplex dual prices associated to these constraints become restricted to take nonnegative values rather than being free real numbers. Let v^*_{DW} be the optimal value of the Dantzig-Wolfe reformulation, and let $\bar{v}_{DW}$ be the LP relaxation value of this formulation. The optimal value of Dantzig-Wolfe reformulation is the same as the optimal value of the original formulation (1)–(5), that is, $v^* = v^*_{DW}$. However, the LP relaxation value of (6), (8)–(9) is better than (or equal to) the LP relaxation value of the original formulation (1)–(5), i.e., $\bar{v} \leq \bar{v}_{DW}$.

The Dantzig-Wolfe reformulation contains a huge number of variables (extreme points), and hence cannot be solved efficiently. Therefore, we do not generate all the feasible columns, but only those that are needed to obtain the optimal Dantzig-Wolfe relaxation value $\bar{v}_{DW}$. In order to generate these columns, we need $\lambda_i \geq 0,\ i \in N$, which is the dual variable of a constraint in (9). The subproblems are used to check if we can find new columns with a negative reduced cost. In this case, the subproblem decomposes into identical problems for each machine, and therefore, we drop the machine index j in the following formulation of the subproblem:

$$\text{Min} \quad z - \sum_{i \in N} \lambda_i x_i \tag{10}$$

$$\text{s.t.} \quad x_i = y_\ell \qquad \forall i \in N, \forall \ell \in L_i \tag{11}$$

$$\sum_{\ell \in L} s_\ell y_\ell = Cz \tag{12}$$

$$x_i, y_\ell, z \in \{0,1\} \qquad \forall i \in N, \forall \ell \in L \tag{13}$$

The objective function (10) minimizes the reduced cost of a new column. Constraints (11)–(13) correspond to our original constraints (3)–(5) for a generic machine. We add new columns to the master as long as we find columns with a negative reduced cost (10). In the above formulation of the subproblem, setting z equal to zero gives us the null solution, where no jobs are assigned to the machine and with a corresponding reduced cost of zero. In practice, we set $z = 1$, and use this formulation to check if we can find a column with a negative reduced cost.

3.2 Lagrangean relaxation for the JGP

The general idea of Lagrangean Relaxation is to delete difficult constraints from the formulation, but re-introduce them in the objective function with a penalty function. Starting from formulation (1)–(5), we relax

the assignment constraint (2) into the objective function, with nonnegative multipliers $\lambda_i \geq 0,\ \forall i \in N$:

$$\text{Min} \quad \sum_{j \in M} z_j - \sum_{i \in N} \lambda_i \left(\sum_{j \in M} x_{ij} - 1 \right) \tag{14}$$

$$\text{s.t.} \quad x_{ij} = y_{\ell j} \qquad \forall i \in N, \forall \ell \in L_i, \forall j \in M \tag{15}$$

$$\sum_{\ell \in L} s_\ell y_{\ell j} = C z_j \qquad \forall j \in M \tag{16}$$

$$x_{ij}, y_{\ell j}, z_j \in \{0,1\} \qquad \forall i \in N, \forall \ell \in L, \forall j \in M \tag{17}$$

This problem decomposes into identical subproblems for each machine j. The problem becomes equivalent to solving m times the same subproblem, where m is the number of machines. To show this, we first re-write the objective function (14) as follows:

$$\sum_{j \in M} z_j - \sum_{i \in N} \lambda_i \left(\sum_{j \in M} x_{ij} - 1 \right) = \sum_{j \in M} \left(z_j - \sum_{i \in N} \lambda_i x_{ij} \right) + \sum_{i \in N} \lambda_i$$

We observe that in this latest expression, the final term is a constant factor. The first term is a summation over all possible machines j of a specific cost function, but the coefficients in this cost function are independent of the machine j. Consequently, the first term is equivalent to solving m times the same subproblem, and we can hence omit the index j for the machines.

$$\sum_{j \in M} \left(z_j - \sum_{i \in N} \lambda_i x_{ij} \right) + \sum_{i \in N} \lambda_i \ = m \left(z - \sum_{i \in N} \lambda_i x_i \right) + \sum_{i \in N} \lambda_i$$

Our Lagrangean problem can hence be re-written as follows:

$$\text{Min} \quad m \left(z - \sum_{i \in N} \lambda_i x_i \right) + \sum_{i \in N} \lambda_i \tag{18}$$

$$\text{s.t.} \quad x_i = y_\ell \qquad \forall i \in N, \forall \ell \in L_i \tag{19}$$

$$\sum_{\ell \in L} s_\ell y_\ell = C z \tag{20}$$

$$x_i, y_\ell, z \in \{0,1\} \qquad \forall i \in N, \forall \ell \in L \tag{21}$$

For optimization purposes, we can ignore the final term in (18) because it is a constant, and we can further leave out the multiplication by m. The objective function in the Lagrangean relaxation formulation hence becomes:

$$\text{Min} \quad z - \sum_{i \in N} \lambda_i x_i \tag{22}$$

Formulations (18)–(21) and (19)–(22) have the same optimal solution. Given the optimal solution and the optimal objective function value (22), we can easily calculate the corresponding value for the original Lagrangean objective function (18). We also note that the subproblem (10)–(13) in the CG procedure is exactly the same as the Lagrangean problem (19)–(22). Hence, be solving the Lagrangean problems, we generate valid assignments (columns) that can be added to the master problem.

Let $\bar{v}_{LR(\lambda)}$ be the optimal solution of the Lagrangean Relaxation formulation for a specific set of multipliers $\lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$. The Lagrangean Dual Problem consists now in finding the multipliers that provides the best bound: $\bar{v}_{LD} = max_{\lambda \geq 0} \bar{v}_{LR(\lambda)}$. The Lagrangean Dual Bound $\bar{v}_{LD}$ is the same as the optimal Dantzig-Wolfe relaxation value $\bar{v}_{DW}$.

Using subgradient optimization, the Lagrangean multipliers are updated in iterative steps. Let $x^r = (x_1^r, x_2^r, \ldots, x_n^r)$ be the optimal assignment variables for the Lagrangean problem (17) – (20) at step r using multipliers $\lambda^r = \{\lambda_1^r, \lambda_2^r, \ldots, \lambda_n^r\}$. The formula used for updating the multipliers is as follows (Fisher 1981):

$$\lambda_i^{r+1} = max\left(0, \lambda_i^r + t^r\left(1 - mx_i^r\right)\right) \qquad \forall i \in N$$

The stepsize t^r is determined as follows:

$$t^r = \frac{\alpha^r\left(UB - \bar{v}_{LR(\lambda^r)}\right)}{\sum_{i \in N}\left(mx_i^r - 1\right)^2} \tag{23}$$

The scalar α^r is set at 2 initially, and reduced by half each time the Lagrangean bound fails to improve for a specific number of iterations. UB represents an upper bound on the Lagrangean Dual value. In a pure Lagrangean Relaxation approach, we set UB equal to the number of available machines, or to a better UB if available. In the combined CG-LR approach, we can take the optimal value of the objective function of the latest restricted master problem.

4 Computational Experiments

In this section, we present the results of our numerical comparison between the regular Column Generation approach (CG) and the combined Column Generation and Lagrangean Relaxation algorithm (CG-LR). First, there are a number of parameters to be set in the subgradient optimization of the Lagrangean Relaxation. Specifically, in the calculation of the step size (23), we have to determine an initial value for α^1 and we have to decide how this step size changes over time. We half the α^r each time the algorithm has failed to improve the Lagrangean lower bound after β iterations. In our testing, we checked the following five combinations of α^1 and β (notation: α^1/β): $2/\infty$, $1/\infty$, $0.5/\infty$, $2/5$, $2/3$. In Figure 2, we see the evolution of the Lagrangean lower bound $\bar{v}_{LR(\lambda)}$ for these various settings on a specific problem instance. The value of the Dantzig-Wolfe LP relaxation (and hence of the Lagrangean Dual) is 3.889, whereas the best setting (2/5) found a Lagrangean lower bound of 3.707. In the remainder of the experiments, we choose the setting 2/5.

Next, we present the results of our testing on a set of 20 instances with 20 jobs and 15 tools and a tool capacity of 8. We compared the Column Generation approach (CG) with the combined Column Generation and Lagrangean Relaxation method (CG-LR). The experiments were done on a workstation with a 2.10 GHz Duo CPU and 2 GB of RAM, using CPLEX 12.2 to solve the master and subproblems. The algorithms were coded within the OPL language of CPLEX. The results are presented in Table 1. The first column indicates the problem instance. The column *LB* gives the final lower bound, obtained at the end of the algorithm. The column *Time* gives the total time to find this lower bound, and it can be subdivided into 3 main elements: the time for the initialization (*INITime*), the time spent solving the master problem (*MPTime*) and the time spent solving the subproblems (*SPTime*). Finally, the column *Iter* gives the total number of iterations, i.e., the number of times that a subproblem is solved. In this implementation of CG-LR, we solve 3 subproblems for each time we solve the master problem. Finally, *MPIter* indicates the number of times the master is solved in CG-LR.

We observe that for this test set, CG-LR finds on average the optimal Dantzig-Wolfe relaxation bound ($\bar{v}_{DW}$) in 126 seconds, compared to 135 for CG. Although this difference is not very large, we observe a more substantial difference when considering the time spent to solve the master problem. The advantage of CG-LR is that we generally have to solve fewer restricted master problems. Indeed, the results indicate that the time spent to solve the master is 3.2 seconds for CG-LR and 14.3 seconds for CG. So we obtain a significant reduction due to the fact that on average we solve only 12 master problems in the CG-LR approach and 39 in the CG approach. However, in total the relative reduction is smaller since most of the time of the algorithm is spent on solving the subproblems (and not the master).

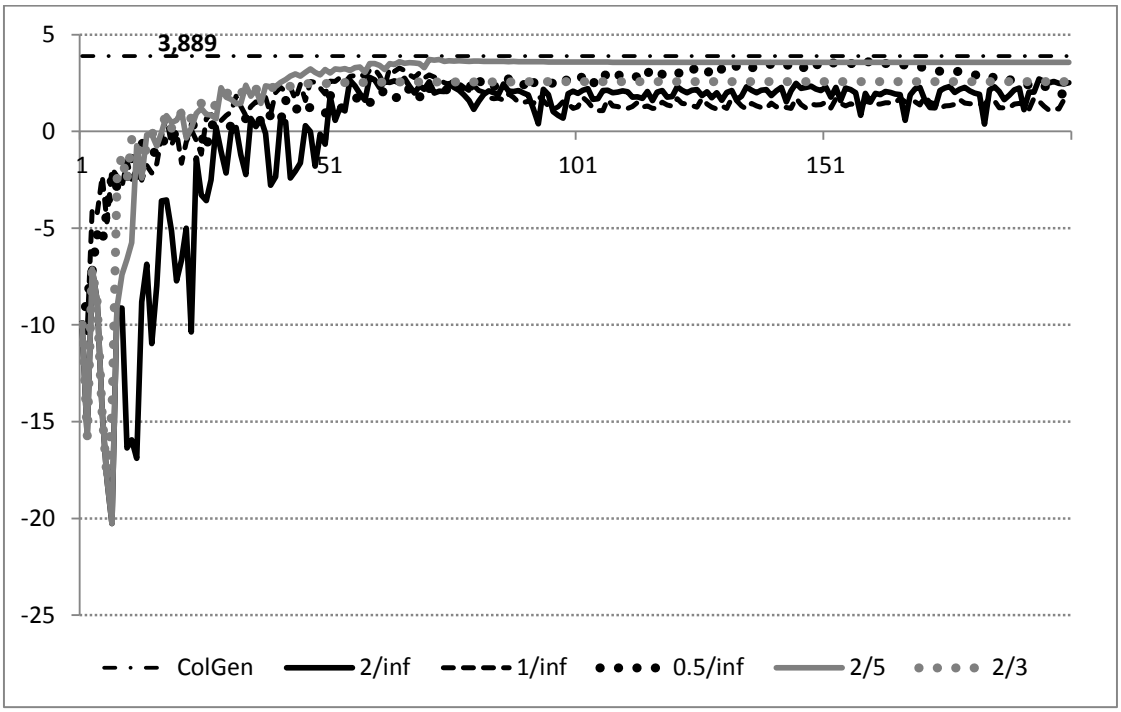

Figure 2: Impact of the LR settings for a specific instance

Table 1: Comparison of the CG and CG-LR approaches

	CG						**CG-LR**						
Prob	LB	Time	INITime	MPTime	SPTime	Iter	LB	Time	INITime	MPTime	SPTime	Iter	MPIter
1	4.50	139.5	0.6	14.5	122.7	47	4.50	133.0	0.2	3.3	129.5	37	13
2	5.50	104.3	0.5	9.4	91.1	32	5.50	122.1	0.5	6.4	115.2	34	12
3	6.00	86.6	1.1	6.1	74.9	25	6.00	106.2	0.6	2.3	103.3	28	10
4	5.88	142.6	0.5	12.0	127.9	41	5.88	161.4	0.1	4.4	156.8	43	15
5	6.25	151.7	0.5	16.2	131.0	43	6.25	87.9	0.5	3.3	84.1	25	9
6	6.25	153.0	0.5	18.3	130.3	43	6.25	84.9	0.5	1.7	82.7	25	9
7	4.50	177.0	0.5	21.0	153.4	51	4.50	137.5	0.6	3.9	133.1	37	13
8	4.00	125.3	0.5	15.5	107.6	44	4.00	121.7	0.5	2.9	118.4	37	13
9	5.88	139.6	0.5	13.5	124.4	39	5.88	157.5	0.5	5.4	151.5	43	15
10	7.00	145.0	0.5	12.4	129.1	38	7.00	151.0	0.5	3.5	147.0	40	14
11	5.00	140.0	0.5	15.1	118.4	40	5.00	127.6	0.6	1.9	125.1	37	13
12	5.36	145.3	1.1	18.1	122.1	42	5.36	146.1	1.1	3.9	141.1	40	14
13	5.50	94.9	0.5	8.2	83.5	29	5.50	78.0	0.5	1.8	75.7	22	8
14	4.90	158.0	0.5	18.2	136.9	43	4.90	162.2	0.6	4.0	157.5	43	15
15	7.00	86.8	0.5	6.6	76.5	25	7.00	65.1	0.5	1.7	62.9	19	7
16	5.75	143.3	0.5	15.0	124.5	39	5.75	146.2	0.6	3.4	142.1	37	13
17	4.00	152.7	0.5	22.4	126.0	52	4.00	136.6	1.1	3.0	132.4	37	13
18	4.73	172.1	0.5	18.5	150.2	46	4.73	137.6	0.5	3.4	133.6	37	13
19	9.00	91.7	1.1	8.5	77.7	24	9.00	97.1	1.1	1.8	94.1	25	9
20	6.38	154.4	0.5	16.8	134.2	44	6.38	170.2	1.0	3.1	166.0	43	15
avg	**5.67**	**135.2**	**0.6**	**14.3**	**117.1**	**39.4**	**5.67**	**126.5**	**0.6**	**3.2**	**122.6**	**34.5**	**12.2**

5 Conclusions

In this paper, we presented and tested a column generation approach which is combined with a Lagrangean Relaxation algorithm. The main advantage of such a combination would be to obtain a better convergence and to solve fewer times the master problem. In our computational experiments, we indeed observed an important reduction by a factor 4 in the number of times we have to solve the master problem. For this

specific problem, most of the time is still spent on solving the subproblems which takes about the same total time in both approaches. Therefore, the reduction in total time is less pronounced.

In many applications of CG, however, a lot of time is spent on solving the master and hence the CG-LR could show a much bigger effect on the overall time. One possible reason why this is not the case for the JGP is that we used the general Branch-and-Bound solver of CPLEX to solve the subproblems, whereas in most applications a dedicated algorithm is used to solve the subproblems. Therefore, we think it is important for further research to also test the CG-LR on problems for which we have a dedicated and fast algorithm to solve the subproblem, and for which hence the most of the total time is spent on the solving the master problem. We will look in the literature for problems that have been solved using CG, but for which solving the master problems takes more time compared to solving the subproblems. An example of these is the class of highly degenerate multiple depot vehicle scheduling problems (Benchimol et al. 2012). Further, it would also be interesting to test other implementations of CG-LR. We could, for example, test the effect of the various factors such as the number of Lagrangean iterations after solving each master problem, the choice of the upper bound, and the updating procedure for the Lagrangean multipliers.

References

Barahona, F., Jensen, D. (1998) Plant location with minimum inventory. *Mathematical Programming* 83, 101–112.

Ben Amor, H., Desrosiers, J., Valerio de Carvalho, J.M. (2006) Dual-optimal inequalities for stabilizing column generation. *Operations Research* 54(3), 454–463,

Benchimol, P., Desaulniers, G., Desrosiers, J. (2012) Stabilized dynamic constraint aggregation for solving set partitioning problems. *European Journal of Operational* Research 223(2), 360–371.

Crama, Y, Oerlemans, A.G. (1994) A column generation approach to job grouping for flexible manufacturing systems. *European Journal of Operational Research* 78, 58–80.

Crama, Y., Oosten, M. (1996) Models for machine-part grouping in cellular manufacturing. *International Journal of Production Research* 34(6), 1693–1713.

Crama, Y., van de Klundert, J., Spieksma, F.C.R. (2002) Production planning in printed circuit board assembly. *Discrete Applied Mathematics* 123, 339–361.

Degraeve, Z., Jans, R. (2007) A new Dantzig-Wolfe reformulation and branch-and-price algorithm for the capacitated lot sizing problem with set up times, *Operations Research* 55(5), 909–920.

Degraeve, Z., Peeters, M. (2003) Optimal integer solutions to industrial cutting-stock problems: Part 2, Benchmark results. *INFORMS Journal on Computing* 15(1), 58–81.

Denizel, M. (2003) Minimization of the number of tool magazine setups on automated machines: A Lagrangean decomposition approach. *Operations Research* 51(2), 309–320.

Desaulniers, G., Desrosiers, J., Ioachim, I., Solomon, M.M., Soumis, F., Villeneuve, D. (1998) A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T.G. Crainic and G. Laporte (eds), *Fleet Management and Logistics*, Kluwer, 57–93.

Fisher, M. (1981) The Lagrangian relaxation method for solving integer programming problems. *Management Science* 27, 1–18.

Geoffrion, A.M. (1974) Lagrangean relaxation for integer programming. *Mathematical Programming Study* 2, 82–114.

Huisman, D., Jans, R., Peeters, M., Wagelmans, A. (2005) Combining column generation and Lagrangian relaxation. In G. Desaulniers, J. Desrosiers, M. Solomon (eds), *Column Generation*, Springer, 247–270.

Konak, A., Kulturel-Konak, S., Azizoğlu, M. (2008) Minimizing the number of tool switching instants in flexible manufacturing systems. *International Journal of Production Economics* 116, 298–307.

Lübbecke, M.E., Desrosiers, J. (2005) Selected topics in column generation. *Operations Research* 53(6), 1007–1023,

Magnanti, T.L., Shapiro, J.F., Wagner, M.H. (1976) Generalized linear programming solves the dual. *Management Science* 22, 1195–1203.

Rajagopalan, S. (1986) Formulation and heuristic solutions for parts grouping and tool loading in flexible manufacturing systems. In K.E. Stecke and R. Suri (Eds.), *Proceedings of the Second ORSA/TIMS conference on Flexible Manufacturing Systems,* Elsevier Science Publishers, Amsterdam, 311–320.

Tang, C.S., Denardo, E.V. (1988) Models arising from a flexible manufacturing machine, Part II: Minimization of the number of switching instants. *Operations Research* 36(5), 778–784.